UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/709,917 | 06/04/2004 | Evan P. Ireland | SYB/0101.01 | 3916 |

31779 7590 04/28/2009

JOHN A. SMART
201 LOS GATOS
SARATOGA RD, #161
LOS GATOS, CA 95030-5308

| EXAMINER | |
|---|---|
| WANG, RONGFA PHILIP | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2191 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 04/28/2009 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

Application Number: 10/709,917
Filing Date: June 04, 2004
Appellant(s): IRELAND, EVAN P.

G. Mack Riddle
For Appellant

**EXAMINER'S ANSWER**

This is in response to the appeal brief filed 2/10/2009 appealing from the Office action mailed 9/12/2008.

**(1) Real Party in Interest**

A statement identifying by name the real party in interest is contained in the brief.

**(2) Related Appeals and Interferences**

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

**(3) Status of Claims**

The statement of the status of claims contained in the brief is correct.

**(4) Status of Amendments After Final**

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

**(5) Summary of Claimed Subject Matter**

The summary of claimed subject matter contained in the brief is correct.

**(6) Grounds of Rejection to be Reviewed on Appeal**

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct except for the 35 U.S.C. 112 rejections of claims 1-46 that has been withdrawn by the examiner.

**(7) Claims Appendix**

The copy of the appealed claims contained in the Appendix to the brief is correct.

**(8) Evidence Relied Upon**

| | | |
|---|---|---|
| 6,182,277 | DeGroot et al. | 1-2001 |
| 5,913,063 | McGurrin et al. | 6-1999 |
| 7,103,885 | Foster | 9-2006 |
| 2003/0055936 | Zhang et al. | 3-2003 |

**(9) Grounds of Rejection**

The following ground(s) of rejection are applicable to the appealed claims:

***Claim Rejections - 35 USC § 103***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

1. Claims 1-5, 15-19, 21-26, 28-30, 33, 40-43, and 45-46 are rejected under 35 U.S.C. 102(b) as being unpatentable over DeGroot et al. (US Patent No. 6,182,277) in view of McGurrin et al. (USPN 5,913,063).

As per claim 1,

DeGroot et al. disclose

- adding a static field of type Component to a program class of the program to create a component (c4: 40-53, "…method of an object…rules may be added to objects of an applicants program, even through the underlying method is compiled…" A method is component in a class and is therefore added.);

- defining at least one attribute specifying declaratively behavior to be added to the program, wherein said at least one attribute comprises active metadata used to generate program code for inclusion in the program (c3: 65-67, "…one or more declarative statements to augment the functionality of at least one method of an object…"; c4: 40-53, "…declarative techniques…that define object behavior…"; c4:

12-30, " The declarative programming technique permit augmentation of methods both on the type system level and on the instance level…Compiled code…compiled to include the extended definitions for the type system…"; c4:49-52, "…rules may be added to objects of an applications program, even through the underlying method is compiled…",where the declarative statements are consider attributes and metadata used to add functionality to an already compiled program code and the augmentation can be applied to existing compiled code by compiling the already compiled program code with extended definition defined in the declarative statements.);

- associating said at least one attribute with the component (c4:1-5, "…associate the declarative statements to ...on the object..."); and

- in response to instantiation of the component at runtime, (c2: 24-25, "…when the method on that object is called…" calling the method instantiates the object.)

DeGroot et al. do not specifically disclose

- generating a subclass based on the program class and said at least one attribute, the subclass including dynamically generated program code based on said at least one attribute;

However, McGurrin et al. disclose

- generating a subclass based on the program class and said at least one attribute, the subclass including dynamically generated program code based on said at least one attribute (c2:27-35, "Some visual coding tools generate source code for object oriented programming languages. With these visual coding tools, a generic object class is typically defined for every type of visual element. When a user draws a particular visual element, the visual coding tool generates source code for (1) a new object class that inherits from the corresponding generic object class, where the attributes of the subclass are initially set to those reflected in the element drawn by a user, and (2) an instance of the new object class. " where user entered attribute and subclass that are generated is described.)

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the teachings of McGurrin into the teachings of DeGroot to include the limitation discloses by McGurrin. The modification would be obvious to one of ordinary skill in the art to want to simplying the process of creating object subclasses based upon existing object classes as suggested by McGurrin et al. (c5: 6-10).

As per claim 2,

the rejection of claim 1 is incorporated,

DeGroot et al. disclose

- wherein said defining step includes defining a particular attribute using active metadata, so as to provide a mechanism for generation of program code from said particular attribute(c2:26-29, "…access to metadata…").

As per claim 3,

the rejection of claim 2 is incorporated,

DeGroot et al. disclose

- wherein said active metadata dynamically generates code for inclusion in a subclass based on the program class(c2:26-29).

As per claim 4,

the rejection of claim 1 is incorporated,

DeGroot et al. disclose

- wherein the generating step includes generating a subclass comprising an instance of a declared component class(c2:15-18, "…new subclass…").

As per claim 5,

the rejection of claim 1 is incorporated,

DeGroot et al. disclose

- Wherein the generating step includes generating a subclass comprising an instance of component class declared as abstract (c4:20-25, "…An abstract specification…").

As per claim 15,

the rejection of claim 1 is incorporated,

DeGroot et al. disclose

- wherein said defining step includes defining attributes for a superclass from which the program class inherits(c1:64-66, "..inheritance…passing attributes…").

As per claim 16,

the rejection of claim 1 is incorporated,

DeGroot et al. disclose

- wherein said defining step includes defining attributes for the program class' package from which the program class inherits(c7: 44-45, "…package…").

As per claim 17,

the rejection of claim 1 is incorporated,

DeGroot et al. disclose

- wherein said defining step includes defining attributes for an interface from which the program class inherits(c2:21-23, "…object interface…").

As per claim 18,

the rejection of claim 17 is incorporated,

DeGroot et al. disclose

- wherein said generating step includes generating an instance of a subclass to mock the behavior of the interface(c2:30-34, "…augment or change…object…").

As per claim 19,

the rejection of claim 1 is incorporated,

DeGroot et al. disclose

- wherein said generating step includes generating code for a non—abstract method body based on an attribute define d for an abstract method(c2:21-25, "…subclassing…pointing to a new function…").

As per claim 21,

the rejection of claim 1 is incorporated,

DeGroot et al. disclose

- adding expected calls as instances of anonymous inner classes of the program; and

  applying runtime introspection by a generated subclass to verify a sequence of expected calls(c12:32-41, "…inner invocation of the actual method…").

As per claim 22,

the rejection of claim 1 is incorporated,

DeGroot et al. disclose

- wherein the component registers itself with a repository when the component is initially activated(c8:5-10, "…object register…").

As per claim 23,

the rejection of claim 22 is incorporated,

DeGroot et al. disclose

- wherein the repository can be queried to determine components that are active(c3:11-15, "…the user may query…objects…").

As per claim 24,

DeGroot et al. disclose

- A computer—readable medium having processor-executable instructions for performing the method of claim 1 ( see rejection of claim 1).

As per claim 25,

DeGroot et al. disclose

- A downloadable set of processor-executable instructions for performing the method of claim 1( see rejection of claim 1)..

As per claim 26,

DeGroot et al. disclose

- (see rejection of claim 1).

As per claim 28,

DeGroot et al. disclose

- the rejection of claim 26 is incorporated, wherein the subclass is a subclass of an abstract class(c4:20-25, "…An abstract specification…").

.

As per claim 29,

the rejection of claim 26 is incorporated,

DeGroot et al. disclose

- wherein said at least one declarative attribute includes active metadata, so as to provide a mechanism for generation of program code(c2:26-29, "…access to metadata…").

As per claim 30,

the rejection of claim 29 is incorporated,

DeGroot et al. disclose

- wherein said active meta— data dynamically generates code for inclusion in the subclass of the program class (c2: 14-25, "…subclassing technique permit…to generate a new method on a subclass..";c8: 6-8, "…may be added at run time..."; c2:40-42, "…the subclassing

technique requires re-compiling the code…”)..

As per claim 33,

the rejection of claim 32 is incorporated,

DeGroot et al. disclose

- wherein the module for generating loads the class containing static attributes before subclass generation (c2: 14-25, “…subclassing technique permit…to generate a new method on a subclass..”;c8: 6-8, “…may be added at run time..."; c2:40-42, “…the subclassing technique requires re-compiling the code…”).

As per claim 40,

the rejection of claim 26 is incorporated,

- See rejection of claim 15.

As per claim 41,

the rejection of claim 26 is incorporated,

- See rejection of claim 17.

As per claim 42,

the rejection of claim 41 is incorporated,

- See rejection of claim 18.

As per claim 43,

the rejection of claim 26 is incorporated,

DeGroot et al. disclose

- wherein the module for generating generates code for a non—abstract system body based on an attribute defined for an abstract method(c4:13-15, "…augmentation…at instance level…").

As per claim 45,

the rejection of claim 26 is incorporated,

- See rejection of claim 22.

As per claim 46,

the rejection of claim 45 is incorporated,

- See rejection of claim 23.

2. Claims 6-14, 20, 31-39, and 44 are rejected under 35 U.S.C. 103(a) as being unpatentable over DeGroot et al. (US Patent No. 6,182,277), McGurrin et al. (USPN 5,913,063) in view of Foster (US Patent No. 7103885).

As per claim 6,

the rejection of claim 1 is incorporated,

DeGroot et al./McGurrin et al. do not specifically disclose

- wherein said defining step includes defining at least one attribute based on comments in source code of the program class.

However, Foster discloses

- wherein said defining step includes defining at least one attribute based on comments in source code of the program class(c7:53-57, "…comment field…a tag relating to an attribute…" Fig. 6 shows comments included in software module and software module is considered source code. The information in tag is considered attribute.).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the teachings of Foster into the teachings of DeGroot et al./McGurrin et al. to include the limitation discloses by Foster . The modification would be obvious to one of ordinary skill in the art to want to process software module based attributes in the comments as suggested by Foster (see col. 2, 2nd paragraph).

As per claim 7,

the rejection of claim 6 is incorporated,

DeGroot et al. disclose

- precompiling a class containing static attributes from said comments(c2:40-42, "…recompiling…").

As per claim 8,

the rejection of claim 7 is incorporated,

DeGroot et al. disclose

- loading the class containing static attributes before subclass generation when a component is instantiated(c2:40-43, "…reloading both the old and new code.").

As per claim 9,

the rejection of claim 6 is incorporated,

DeGroot et al. disclose

- defining an automated mapping between attribute syntax in comments and attribute syntax as expressed in generated program code(c3:40-45, "…maps…").

As per claim 10,

the rejection of claim 1 is incorporated,

Foster discloses

- Said step defining step includes defining at least one attribute in a property file external to the program class(c5:35-38, "…management file…attribute…").

As per claim 11,

the rejection of claim 10 is incorporated,

Foster discloses

- compiling a class containing dynamic attributes from said property file(c5:35-38, "…management file…attribute…").

As per claim 12,

the rejection of claim 11 is incorporated,

DeGroot et al. disclose

- loading the class containing dynamic attributes before subclass generation when a component is instantiated(c2:40-42, "…recompiling…").

As per claim 13,

the rejection of claim 10 is incorporated,

DeGroot et al. disclose

- defining an automated mapping between attribute syntax in a property file and attribute syntax as expressed in generated program code(c3:40-45, "…maps…").

As per claim 14,

the rejection of claim 10 is incorporated,

DeGroot et al. disclose

- wherein attributes in the property file comprise property name and property value pairs(c6:35-38, "…values of parameters…").

As per claim 20,

the rejection of claim 1 is incorporated,

Foster discloses

- wherein said generating step includes generating program code based on comments in a source file(c7:53-57, "…comment field…a tag relating to an attribute…").

As per claim 31,

the rejection of claim 26 is incorporated,

- wherein the attribute module provides for defining at least one attribute based on comments in source code of the program class(see rejection of claim 6).

As per claim 32,

the rejection of claim 31 is incorporated,

a precompiler for precompiling a class containing static attributes from said comments().

As per claim 34,

the rejection of claim 31 is incorporated,

- see rejection of claim 13.

As per claim 35,

the rejection of claim 26 is incorporated,

- See rejection of claim 10.

As per claim 36,

the rejection of claim 35 is incorporated,

- see rejection of claim 12.

As per claim 37,

the rejection of claim 36 is incorporated,

DeGroot et al. disclose

- wherein the module for generating loads the class containing dynamic attributes before subclass generation when a component is instantiated (c2: 14-25, "…subclassing technique permit…to generate a new method on a subclass..";c8: 6-8, "…may be added at run time..."; c2:40-42, "…the subclassing technique requires re-compiling the code…").

As per claim 38,

the rejection of claim 35 is incorporated,

DeGroot et al. disclose

- an automated mapping between attribute syntax in a property file and attribute syntax as expressed in generated program code(c3:40-45, "…maps…").

As per claim 39,

the rejection of claim 35 is incorporated

- See rejection of claim 14.

As per claim 44,

the rejection of claim 26 is incorporated,

- See rejection of claim 20.

3. Claims 47-61 are rejected under 35 U.S.C. 103(a) as being unpatentable over DeGroot et al. (US Patent No. 6,182,277) in view of McGurrin et al. (USPN 5,913,063) and further in view of Zhang et al. (US Patent No. 2003/0055936).

As per claim 47,

DeGroot et al. disclose

- defining at least one attribute specifying declaratively behavior which is desired to be added to an application without access to the application source code (c4: 40-53, "…declarative techniques…that define object behavior…"; c3: 65-67, "…one or more declarative statements to augment the functionality of at least one method of an object…"; c4:49-52, "…rules may be added to objects of an applications program, even through the underlying method is compiled…" which means the source code is not available.);

- wherein said at least one attribute comprises active metadata used to generate code adding behavior to the

application(c4: 40-53, "…declarative techniques…that define object behavior…"; c4: 28-32, "…compiles to include the extended definitions for the type system…");

- compiling the application and the attributes(c4: 28-32, "…compiles to include the extended definitions for the type system…");

- adding behavior to the application based on said at least one attribute (c3: 65-67, "…one or more declarative statements to augment the functionality of at least one method of an object…").

DeGroot et al. do not specifically disclose

- and generating a subclass which includes dynamically generated code,

However, McGurrin et al. disclose

- and generating a subclass which includes dynamically generated code,(c2:27-35, "Some visual coding tools generate source code for object oriented programming languages. With these visual coding tools, a generic object class is typically defined for every type of visual element. When a user draws a particular visual element, the visual coding tool generates source code for (1) a new object class that inherits from the corresponding generic object class, where the attributes of the subclass are initially set to those reflected in the element drawn

by a user, and (2) an instance of the new object class. “, where user entered attribute and subclass that are generated is described.)

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the teachings of McGurrin into the teachings of DeGroot to include the limitation discloses by McGurrin. The modification would be obvious to one of ordinary skill in the art to want to simplying the process of creating object subclasses based upon existing object classes as suggested by McGurrin et al. (c5: 6-10).

DeGroot et al..McGurrin et al. do not specifically disclose

- storing said at least one attribute in a properties file external to the application; creating a dynamic attributes class based on the properties; compiling the application and the dynamic attributes class;

However Zhang et al. disclose

- storing said at least one attribute in a properties file external to the application; creating a dynamic attributes class based on the properties file; compiling the dynamic attributes class; ([0072], “…define a new dynamic attribute class…compile the dynamic attribute class…”; [0075], “…the dynamic attribute list is a text file for...each dynamic attribute class..." the text file is a properties file external to the application. ).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the teachings of Zhang et al. into the teachings of DeGroot et al..McGurrin et al. to include the limitation discloses by Zhang et al. . The modification would

be obvious to one of ordinary skill in the art to want to be able to define new dynamic attributes as suggested by Zhang et al. ([0013]).

As per claim 48,

the rejection of claim 47 is incorporated,

- DeGroot et al. disclose

  wherein said defining step includes defining a particular attribute using active metadata, so as to provide a mechanism for generation of program code from said particular attribute(c2:26-29, "…access to metadata…").

As per claim 49,

the rejection of claim 48 is incorporated,

- see rejection of claim 3.

As per claim 50,

the rejection of claim 47 is incorporated,

DeGroot et al. disclose

- wherein said defining step includes defining an attribute for overriding a method of the application(c2:50, "…method overriding…").

As per claim 51,

the rejection of claim 47 is incorporated,

DeGroot et al. disclose

- wherein said defining step includes defining an attribute for extending a method of the application(c4:26-30, "…extended definition…").

As per claim 52,

the rejection of claim 47 is incorporated,

DeGroot et al. disclose

- loading the class containing dynamic attributes before generating the subclass(c12:36-40, "…dynamically loaded…").

As per claim 53,

the rejection of claim 47 is incorporated,

DeGroot et al. disclose

- defining an automated mapping between attribute syntax in the properties file and attribute syntax as expressed in generated program code(c3:40-45, "…maps…").

As per claim 54,

the rejection of claim 47 is incorporated,

DeGroot et al.

- attribute in the properties file comprise property name and property value pairs(c6:35-38, "…values of parameters…").

As per claim 55,

the rejection of claim 47 is incorporated,

DeGroot et al. disclose

- wherein said creating step includes creating a dynamic attributes class using a pre— compiler(c2:40-42, "…recompiling…").

As per claim 56,

the rejection of claim 47 is incorporated,

Zhang et al. disclose

- wherein said creating step includes creating a dynamic attributes class at runtime([0070], "…creating of dynamic attribute classes…").

As per claim 57,

the rejection of claim 47 is incorporated,

Zhang et al. disclose

- wherein said compiling step includes using a Java compiler (JAVAC)([0030], "…Java classes are compiled…".)

As per claim 58,

the rejection of claim 47 is incorporated,

DeGroot et al. disclose

- wherein said generating step includes using a precompiler(c2:40-42, "…recompiling…").

As per claim 59,

the rejection of claim 47 is incorporated,

Zhang et al.

- wherein said generating step includes using a runtime compiler(c2:40-42, "…recompiling…").

. As per claim 60,

- see rejection of claim 47.

As per claim 61,

- see rejection of claim 47.

As per claim 27,

the rejection of claim 26 is incorporated,

DeGroot et al. do not specifically disclose

- wherein the subclass adds tracing behavior to a program.

However Zhang et al. disclose

- wherein the subclass adds tracing behavior to a program([0051], "…debug tracing…").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the teachings of Zhang et al. into the teachings of DeGroot et al. to include the limitation discloses by Zhang et al. . The modification would be obvious to one of ordinary skill in the art to want to be able to define new dynamic attributes as suggested by Zhang et al. ([0013]).

**(10) Response to Argument**

Response to argument A :

Upon further review, the examiner withdraws the 35 U.S.C. 112 rejections of claims 1-46.

The objection to the specification has been withdrawn in view of the appellant's amendment to the specification removing hyperlinks.

Response to argument B:

***B1) The Appellant argued --***

"attribute" in "defining at least one attribute specifying declaratively behavior to be added to the program" as a language construct that developers can use.

***B1) Examiner's Response –***

The examiner wants to point out that there is no language in claim 1, for example, including the limitation that attribute is a language construct. Such argument can be considered as out of the scope of the claim language. Further, in one interpretation, an attribute is considered as information that can be used to specify declaratively behavior to be added to the program. DeGroot, c3:65-67, "...submits one or more declarative statements to augment the functionality of at least one method of on object…", clearly shows that declarative statements being used as attributes to specify declaratively behavior to be added to the program. The Appellant additionally argued "rules" of DeGroot are fundamentally different from attributes. As explained above, information that can be used to specify declaratively behavior is considered as attribute. The declarative statements or rules of DeGroot are considered as information used to specify declaratively to be added to the program. For reasons presented above, the examiner considers DeGroot clearly discloses attribute as claimed.

***B2) The Appellant argued --***

The recited prior art does no disclose one attribute comprises active metadata used to generate code.

***B2) Examiner's Response –***

Per specification, [0126], line 4, "attribute can be considered as active metadata". For at least the reasons above, DeGroot discloses attribute and therefore active metadata. In claim 1, for example, claims “active metadata is used to generate program code”. DeGroot’s declarative statements are used to augment functionality of objects and therefore are considered metadata used to generate program code.

***B3) The Appellant argued --***

The recited prior art does not disclose dynamically generate additional code to extend program functionality.

***B3) Examiner’s Response –***

As previously discussed, DeGroot discloses using declarative statements as attributes to augment the functionality of a program (c3:65-67). DeGroot further discloses such augmentation can be applied at type system level or instance level, see c4:13-18, "The declarative programming technique permits augmentation of methods both on the type system level and on the instance level…to associate declarative statements to a method on a specific instance of the object, or…all instances of the object”. In object-oriented programming terms, such augmentation can be associated at class level (type system level) or object instance level. DeGroot, c427-29, further discloses, “Compiled code…and compiled to include the extended definitions for the type system…” and c4:48-52, “…rules may be added to objects of an applications program, even though the underlying method is compiled..." further discloses generating code with compilation of existing code and the declarative statement to extend program functionality. For reasons above, the examiner consider DeGroot discloses dynamically generate additional code to extend program functionality.

***B4) The Appellant argued --***

The Appellant also argued McGurrin fails to teach or suggest dynamically generating program code based on defined attributes.

***B4) Examiner's Response –***

Referring to response in previous paragraphs, the examiner has shown at least DeGroot discloses dynamically generating program code and therefore the combination of DeGroot and McGurrin discloses dynamically generating program code based on defined attributes.

For reasons presented above, the examiner considers the combined references discloses or teaches the argued limitation.

Response to argument C:

***C1) The Appellant argued --***

The recited prior art does not provide any teaching of "defining attributes comprising active metadata used to generate code for adding behavior to a program as it executes at runtime"

***C1) Examiner's Response –***

Per specification, [0126], line 4, "attribute can be considered as active metadata". For at least the reasons above, DeGroot discloses attribute and therefore active metadata. In claim 1, for example, claims "active metadata is used to generate program code". DeGroot's declarative statements are used to augment functionality of objects and therefore are considered metadata used to generate program code.

***C2) The Appellant argued --***

Regarding Appellant's argument that the reference provides no teaching or suggestion of attributes defined in source code comments of the program itself.

***C2) Examiner's Response –***

The Appellant argued that Foster's attribute is not comparable with the Appellant's invention because "An attribute class is generated based on these comments...(page 19 of the brief, 1st paragraph)" The examiner wants to point out that there is no language including an attribute class being generated in claim 6. Such argument is considered out of the scope of the claim language of claim 6. Foster c7:54-58, discloses "…comment field of a version file system, a tag relating to an attribute of the associated software module" The tag in comment is considered an attribute associated with a software module. Fig. 6 shows a software module including a comment field to define attributes. Therefore, the examiner considers the recited prior art discloses the argued limitation.

For reasons presented above, the examiner considers the combined references discloses or teaches the argued limitation.

Response to argument D:

***D1) The Appellant argued --***

Per Appellant argument that Zhang's attributes do not comprise active metadata for adding behavior to program code.

***D1) Examiner's Response --***

Please refer to response to argument B2 of examiner's response related to DeGroot disclosing attributes comprising active metadata for augmentation of program code. The examiner considers the recited prior art discloses such limitation.

***D2) The Appellant argued --***

Appellant argued the combined references does not discloses compiling the application and the dynamic attributes classes so as to generate a subclass which dynamically generated code adding behavior to the application.

***D2) Examiner's Response --***

Referring to examiner's response to argument B, DeGroot discloses defining attributes (c3:65-67, "…one or more declarative statements to augment the functionality of al least one method of an object) and compiling application and attributes to dynamically generate code adding behavior to the application (c4:c4:27-30, "Compiled code...and compiled to include the extended definitions for the type system"). McGurrin et al. disclose generating a subclass which includes dynamically generated code,(c2:27-35, "Some visual coding tools generate source code for object oriented programming languages. With these visual coding tools, a generic object class is typically defined for every type of visual element. When a user draws a particular visual element, the visual coding tool generates source code for (1) a new object class that inherits from the corresponding generic object class, where the attributes of the subclass are initially set to those reflected in the element drawn by a user, and (2) an instance of the new object class. ", where user entered attribute and subclass that are generated is described.) The combination of DeGroot and McGurrin discloses compiling the application and dynamic attributes so as to generate a subclass which dynamically generate code adding behavior to the application. Further Zhang disclose compiling the dynamic attributes class to generate code; ([0072], "…a

dynamic attribute class…compile the dynamic attribute class…"). Therefore, the combination of DeGroot, McGurrin and Zhang discloses the argued limitations.

For reasons presented above, the examiner considers the combined references discloses or teaches the argued limitation.

**(11) Related Proceeding(s) Appendix**

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Philip R. Wang /Philip R. Wang/ 4/23/2009

Conferees:

Wei Zhen
/Wei Y Zhen/
Supervisory Patent Examiner, Art Unit 2191

Lewis A. Bullock, Jr.
/Lewis A. Bullock, Jr./
Supervisory Patent Examiner, Art Unit 2193

Philip R. Wang /Philip R. Wang/ 4/23/2009